Use of Dynamic Programming for the Design of Irregular LDPC Codes

Sang Hyun LEE[†], Duho RHEE[‡], Il Mu BYUN[‡], and Kwang Soon KIM[‡]

[†] Mobile Communications Lab.
Electronics and Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, Korea
E-mail: sh-lee@etri.re.kr

[‡]Dept. of Electrical and Electronic Engineering
Yonsei University
134 Sinchon-dong, Seodaemun-gu, Seoul, Korea
E-mail: ks.kim@yonsei.ac.kr

Abstract

A simple design method using dynamic programming is proposed for good LDPC codes with relatively low code-rate. By applying a dynamic programming optimization to the construction of the portion fixed for easy encoding in the parity-check matrix, we can maximize the girth associated with columns contained in that portion of the matrix. Simulation results show performance improvement over the conventional controlled random construction method.

1. INTRODUCTION

Recently, LDPC codes have been employed in a variety of practical applications including mobile communication applications with link adaptation feature, in which a suite of various channel codes is required [1]. Although such codes have different code-rates and codeword-lengths, it is practically preferred that, for easy encoding, their corresponding parity-check matrices contain a triangular sub-structure [2], which will be referred to as an echelon form. The echelon form is a lower triangular sub-matrix with its main diagonal and the first diagonal below the main diagonal, which will referred to as the first diagonal, filled with nonzero entries. Also, it is commonly comprised of the low-degree columns in the parity-check matrix because the triangulated sub-structure is reserved for the parity-check bits of codewords. In [3], a special echelon form comprised of only degree-two columns was proposed to keep the minimum distance and the girth as large as possible. However, various techniques to evaluate the optimal degree distribution including the density evolution usually lead to more occurrence of columns with degree of more than two in the parity-check columns as the code rate decreases [4]. Thus, the echelon form may contain columns with degrees of more than two and could not be determined in the trivial form as in [3]. Even in this case, however, by applying an echelon form in the parity-check matrix, several diagonals are fixed in advance and only a small portion of off-diagonals needs to be configured. In addition, we can easily evaluate the girth of the code because the lengths of cycles contained in the Tanner graph associated with the parity-check matrix can be straightforwardly determined by evaluating the 'distance' between nontrivial entries in the echelon form. Although such a sub-structure with degrees of more than two could not exploit the simple encoding property in [3], it allows easy encoding by back-substitution still with the computational complexity of $O(n)$. In particular, for low-rate codes, most part of the parity-check matrix is occupied by the echelon form. Thus, the number of entries to be configured by a controlled random construction, such as the PEG algorithm [5], is reduced and the decoding performance can be managed by the configuration of the sub-structure. Since an implicit estimation of the decoding performance is available with the distribution of cycles, the locations of configurable entries can be determined in order to maximize the length of associated cycles. Although such an optimization problem is readily seen to be $\mathcal{NP}$-hard, we relax the problem so that each additional entry is placed in a sequential manner. This paper addresses the design method of a parity-check matrix with the echelon form using dynamic programming.

2. PROPOSED ALGORITHM

For an LDPC code of our interest, the parity-check matrix of the code, $H = [H_1\ H_{tr}]$, is partitioned into two sub-structures, where H_{tr} is an echelon form defined as above and denotes a sub-matrix containing all parity-check columns, and H_1 is a random sub-matrix designed by a controlled random construction, such as the PEG algorithm[5]. Furthermore, H_{tr} does not need to be a 'square' triangular matrix with the rightmost column of degree one, but could be any 'rectangular' triangular matrix whose number of columns is less than that of rows. For the sake of easy understanding, we

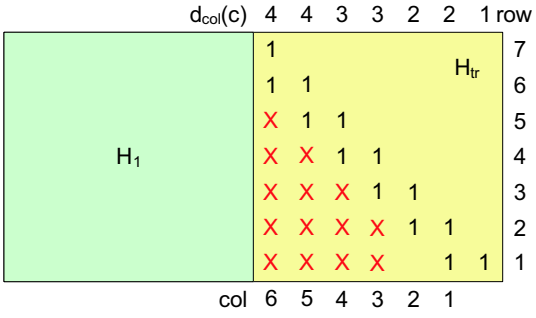

Figure 1: The representation of the echelon form.

will explain the scheme with a simple example. However, the following assertions are valid for any parity-check matrix whose right part is an echelon form. Fig. 1 shows the parity-check matrix with the echelon form of seven rows and columns. Here, $d_{col}(\mathbf{c})$ is a sequence of the degrees of columns contained in the echelon form. Then, a column c has $u(d_{col}(c)-2)$ remaining entries which have not yet been placed, where $u(x) = 0$ if $x \leq 0$, and $u(x) = x$ if $x > 0$. The $\times$'s in each column of H_{tr} represent available locations for the remaining entries. Note that available locations are restricted to the lower-triangular portion in order to maintain the easy encoding feature. The indices of rows and columns, *row* and *col*, denote the locations of the $\times$'s in H_{tr}, respectively. Since the column in the right portion of the echelon form is more likely to cause a smaller cycle, the configuration of new entries in H_{tr} commences at the rightmost column with at least one remaining entry to be placed. Since the locations of new entries are closely related to lengths of cycles containing those entries, the examination of cycles is performed for all available locations of each column with a proper metric and the evaluated metrics will be compared to determine the best location. As the configuration goes on, the addition of a new entry causes additional cycles formed among the new entry and previously located entries up to the current column. Here, the metric can be separated into three parts caused by: i) cycles formed between the new entry and the diagonal entries, ii) cycles formed between new entry and the placed entries up to the previous column, and iii) cycles formed among new entries in the same column and the diagonal entries., and iv) the sum of metrics of the entries which have been placed up to the previous column. By comparing the metrics for all possible locations for the current entry, the best location is determined. Such a procedure is then repeated until locations of all entries are determined. To evaluate the metric function, the distribution of cycles should be examined. By a brief glance, no new entry can avoid forming a cycle containing it and the length of such a cycle is determined by the location of the new entry. Here, we add one more constraint that additional entries in columns with at least two additional entries should not have the same row location. This constraint guarantees that cycles formed among added entries are avoided. Then, there exist three types of cycle caused by addition of a new entry: the one formed by the new entry and nonzero entries on two diagonals, another one among the new entry and previously located entries, and the other among more than two new entries in the same column. For the mth new entry of the lth column, $\mathbf{e}_m^l$, the length of the first type cycle, $L_1(\mathbf{e}_m^l)$, is determined by

$$L_1(\mathbf{e}_m^l) = 2(\Delta_r(\mathbf{e}_m^l) + 1). \quad (1)$$

Here, $\Delta_r(\mathbf{e}_m^l)$ is the absolute value of the difference between the row location of $\mathbf{e}_m^l$ and the nonzero entry in the same column on the first diagonal. Let $L_2(\mathbf{e}_m^l, \mathbf{e})$ denote the length of the second type cycle caused by new entry $\mathbf{e}_m^l$ and a previously placed entry $\mathbf{e}$. Then, $L_2(\mathbf{e}_m^l, \mathbf{e})$ is given by

$$L_2(\mathbf{e}_m^l, \mathbf{e}) = 2(\Delta_r(\mathbf{e}_m^l, \mathbf{e}) + \Delta_c(\mathbf{e}_m^l, \mathbf{e}) + 1), \mathbf{e} \in E_p. \quad (2)$$

Here, $\Delta_r(\mathbf{e}_1, \mathbf{e}_2)$ and $\Delta_c(\mathbf{e}_1, \mathbf{e}_2)$ are the absolute values of the differences between the row locations and between the column locations of $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively. Furthermore, E_p denotes the set of all previous entries that have been placed (the entries on the two diagonals are excluded). Finally, let $L_3(\mathbf{e}_m^l, \mathbf{e}_{m'}^l)$ be the length of the third type cycle caused by the new entry $\mathbf{e}_m^l$ and an entry $\mathbf{e}_{m'}^l$ added (simultaneously) in the same column. Then, $L_3(\mathbf{e}_m^l, \mathbf{e}_{m'}^l)$ is calculated by

$$L_3(\mathbf{e}_m^l, \mathbf{e}_{m'}^l) = 2(\Delta_r(\mathbf{e}_m^l, \mathbf{e}_{m'}^l) + 1), \mathbf{e}_{m'}^l \in E_c \setminus \{\mathbf{e}_m^l\}, \quad (3)$$

where E_c is the set of all entries simultaneously placed in the current column. Then, the metric function for the distribution of cycles caused by the addition of $\mathbf{e}_m^l$ is defined by

$$M(\mathbf{e}_m^l) = B^{-\frac{1}{2}L_1(\mathbf{e}_m^l)} + \sum_{\mathbf{e}\in E_p} B^{-\frac{1}{2}L_2(\mathbf{e}_m^l,\mathbf{e})} + \sum_{\mathbf{e}\in E_c\setminus\{\mathbf{e}_m^l\}} (B^{-\frac{1}{2}L_3(\mathbf{e}_m^l,\mathbf{e})} + B^{-\frac{1}{2}\Lambda_{E_p}(\mathbf{e}_m^l,\mathbf{e})}) \quad (4)$$

Here, a function $\Lambda_{E_p}(\mathbf{e}_1, \mathbf{e}_2), \{\mathbf{e}_1, \mathbf{e}_2\} \in E_c$ is defined by

$$\Lambda_{E_p}(\mathbf{e}_1, \mathbf{e}_2) = \begin{cases} 4 & {}^{\exists}\mathbf{e} \in E_p \text{ such that} (\mathbf{e})_r \in \{(\mathbf{e}_1)_r, (\mathbf{e}_2)_r\}, \\ & (\mathbf{e})_c = \max((\mathbf{e}_1)_r, (\mathbf{e}_2)_r) - l, l = 0, 1 \\ \infty & \text{otherwise}, \end{cases} \quad (5)$$

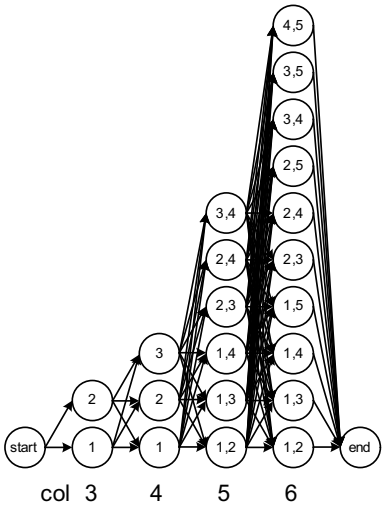

Figure 2: The trellis for the dynamic programming.

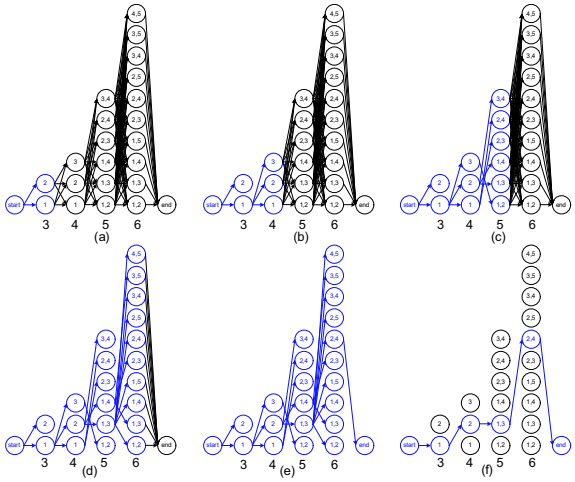

Figure 3: An example of dynamic programming procedure.

where $(\mathbf{e})_c$ and $(\mathbf{e})_r$ denote the column and the row locations of $\mathbf{e}$, respectively. $\Lambda_{E_p}(\mathbf{e}_1, \mathbf{e}_2)$ is introduced to deal with a special case that a column with only one additional entry forms a cycle between the current column which has at least two additional entries. The exponential representation of the metric function in (4) is justified by the observation that a small number of short cycles are even more likely to impair the decoding performance than a large number of large cycles. Note that a single cycle of length $2L$ has the same impact as B cycles of length $2(L+1)$ in this metric. Thus, B should be carefully chosen to quantify the relation between adjacent length of cycles.

Now, consider the construction of a trellis for dynamic programming. Fig. 2 depicts the trellis for the echelon form of the parity-check matrix in Fig. 1. Here, the *start* and *end* nodes indicate the origination and the termination of the trellis, respectively. Several columns of nodes between two trivial nodes enumerate all possible configurations of the additional entries for the corresponding column in the echelon form. The digit labelled below each column in the trellis indicates the column location in the parity-check matrix, as defined in Fig. 1. The digits labelled on each node represent the row locations of new entries. The number of nodes on each column depends on both the number of locations available for the new entries and the number of remaining entries to be placed. For the lth column with N_l available locations and M_l remaining entries to be placed, $\binom{N_l}{M_l}$ configurations are possible. For practical applications such as in [1], M_l is usually set to a small number. Thus, each column with $\binom{N_l}{M_l}$ nodes in the trellis may be treated with manageable complexity. Edges emanating from a node of one column and arriving at each node of the adjacent column denote all possible configurations for entries of the next column after the addition at the current column. Although the connectivity of edges is controlled to cover all possible configurations of the entry addition, some edges will be obviated by a pruning procedure to meet the constraint noted above. To do so, we impose on the connection of edges a constraint that, for a column with at least two entries to be added, the set of entries in the current column has no common entry with the previous columns with at least two added entries. The constraint assures that all cycles formed by only added entries are avoided. The procedure of the entry addition employs the conventional dynamic programming [6], except for the pruning procedure of some nodes. The pruning procedure is followed by the selection of the edge with the best metric up to current column among arriving edges at the node. For a given node labelled with at least two digits in the current column, all nodes in the previous column are connected to it by the corresponding edges. Also, such nodes in the previous column have the corresponding survived paths that contain the information of nodes selected up to that column. If any node labelled with at least two digits in the survived path has an identical row location with the entry at the node in the current column, the corresponding edge is discarded before the selection of the best metric. This operation avoids all cycles formed among added entries. After the pruning procedure, the edge with the best metric up to current column is selected among arriving edges at each node while the remaining edges are discarded. Fig. 3 shows an example of the addition procedure with the trellis associated with the echelon form in Fig. 1. In (a), the

procedure is initialized by evaluating the metrics for all first type cycles formed by the first new entry and diagonal entries. In (b)-(d), each node in the column chooses the path associated with the best metric value. At the last step ((e)), the path associated with the best metric is chosen among all possible paths to complete the addition of new entries. Finally, the resultant entry configuration can be represented by the indices of the rows and columns as

$$\{(\mathbf{r}, \mathbf{c})\} = \{(1,3), (2,4), (1,5), (3,5), (2,6), (4,6)\}. \quad (6)$$

The remaining part of the parity-check matrix, H_1, can then be constructed from the initial graph associated with H_{tr} in a controlled random manner, which requires additional cycle conditioning scheme. However, since most part of the parity-check matrix has been already designed, the design of H_1 becomes simple.

3. DESIGN EXAMPLE

We used the proposed structure to design simple LDPC codes with code rate 1/3 and 1/4. The column degree profile is given as $\{2, 4, 20\}$ and the distribution of the degrees is chosen for each code rate by using the density evolution technique [4] with the constraint that the number of degree-two columns in the parity-check matrix is less than half the number of entire rows. We set B in (4) to 10. In Fig. 4, the performance of the constructed codes is shown over AWGN channel. The number of iteration for the message-passing decoding is 100. Here, the conventional code denotes the LDPC code containing an echelon form and using the same degree distribution but constructed by the PEG algorithm with girth conditioning. From Fig. 4, the performance of the proposed code is better than the conventional code. This is mainly due to the fact that the lengths of overall cycles formed with the low degree columns are considered in the proposed algorithm while the conventional algorithm only guarantee the minimum length of cycle. The simulation result shows that, at given code rate and degree distribution, the performance improvement can be achieved by allowing a particular structure. As the degrees of the columns increase, the complexity of the proposed algorithm grows very fast. However, for a code with relatively short codeword-length, the proposed algorithm runs a little bit faster than the PEG algorithm because of no need to the actual evaluation of cycles.

4. CONCLUSIONS

In this paper, an LDPC code structure using an echelon form and its design method using dynamic programming were proposed for the design of low-rate LDPC codes. Compared to the conventional controlled random construction, the proposed method eliminates small cycles associated with low-degree columns more effectively since the proposed code structure greatly simplifies the evaluation of the girth. In addition, the proposed structure can adopt an optimized degree distribution while maintaining the linear encoding complexity. Simulation results showed that the proposed code achieves performance improvement over the conventional codes.

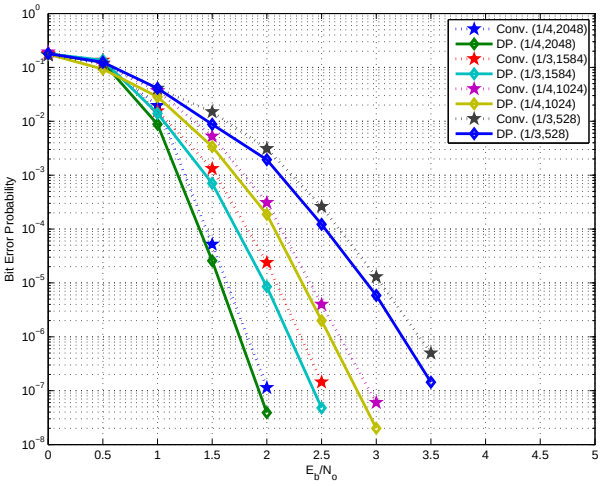

Figure 4: The performance of designed codes over AWGN channel

References

[1] B. Classon *et al.*, "LDPC coding for OFDMA PHY," *IEEE C802.16e-05/066r3*, January 2005.

[2] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638-656, February 2001.

[3] M. Yang, W.E. Ryan, and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, pp. 564-571, April 2004.

[4] http://lthcwww.epfl.ch/research/ldpcopt/

[5] X. Hu, E. Eleftheriou, and D. Arnold, "Progressive edge-growth Tanner graphs," *Proc. IEEE GlobeCom*, vol. 2, pp. 995-1001, San Antonio, TX. U.S.A., November 2001.

[6] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.